**Task: For each sentence identify other similar sentences**

# Loading Dependencies

In [104]:
```
import os
import re
import heapq
import pickle
import numpy as np
from tqdm import tqdm
from scipy.spatial.distance import cdist
import pandas as pd

from nltk import pos_tag
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
from nltk.corpus import wordnet as wn
from nltk.stem.wordnet import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()

from sklearn.decomposition import PCA
from sklearn.feature_extraction.text import TfidfVectorizer
# import xlrd
# import openpyxl
```

In [2]:
```
DATA_DIR = "data"
INPUT_FILE = "Interview_Identify_similar_sentences.xlsx"
LEXICAL_OUTPUT_FILE = "Interview_Identify_similar_sentences_output.xlsx"
```

# Data Loading

In [43]:
```
input_path = os.path.join(DATA_DIR, INPUT_FILE)
df = pd.read_excel(input_path, header = None, names=['sentence'])
df.shape
```

Out[43]: (9534, 1)

In [44]:
```
# Adding an additional ID column
df.insert(0, 'sentenceId', np.arange(df.shape[0]))
```

In [46]:
```
df.sample(10)
```

Out[46]:

| | sentenceId | sentence |
|---|---|---|
| **3969** | 3969 | How do you know who viewed you video on Instag... |
| **9507** | 9507 | How do I stop caring for people who don't real... |
| **9279** | 9279 | What,Äôs your biggest regret that you have in ... |
| **9056** | 9056 | What is a hard disk? What function does it hav... |
| **5318** | 5318 | What knowledge should one have to create an op... |
| **2164** | 2164 | How can we know that the Illuminati is real? |
| **8202** | 8202 | How do you have the motvitation to exercise? |
| **2988** | 2988 | Do you have any tips for coping with anxiety? |
| **6538** | 6538 | Can a boy join the Indian armed forces after a... |
| **3335** | 3335 | How can I get 99 percentile and above in CAT 2... |

# Text Preprocessing

In [47]:
```
# List all the characters in entire data
print(sorted(set(' '.join(df['sentence'].values.tolist()))))
```

```
[' ', '!', '"', '#', '$', '&', "'", '(', ')', '*', '+', ',', '-', '.', '/', '0
', '1', '2', '3', '4', '5', '6', '7', '8', '9', ':', '<', '>', '?', '@', 'A',
'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q
', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', '[', '\\', ']', '^', '_', '`',
'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p
', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '|', '¥', '©', '¬', '°',
'Ä', 'Ç', 'Ô', 'ì', 'ò', 'ô', 'ù', 'ú', 'ü', 'π', ',', '†', '√']
```

Perform below steps for cleaning

- **Add Extra space before and after .,!?**
- **Except for alphabets(lower & capital), numbers, .,!'? replace other characters with spaces**
- **Remove extra and trailing spaces**

In [48]:
```
def clean_text(text):
    # add extra space before and after .,!?
    pattern = r"([.,!?])"
    text = re.sub(pattern, r" \1 ", text)
    # replace extra characters with space
    pattern = r"[^a-zA-Z0-9.,!'?]+"
    text = re.sub(pattern, " ", text)
    # Remove extra spaces
    pattern = r"\s+"
    text = re.sub(pattern, " ", text)
    return text.strip()
```

In [49]:
```
# Sample Test
clean_text("What does Balaji Vishwanathan think about Modi's new currency Ide
a?")
```

Out[49]:
```
"What does Balaji Vishwanathan think about Modi's new currency Idea ?"
```

In [50]:
```
# apply preprocess steps on all records
df['clean_sentence'] = df['sentence'].apply(clean_text)
df.sample(5)
```

Out[50]:

| | sentenceId | sentence | clean_sentence |
|---|---|---|---|
| **8101** | 8101 | What is the biggest/most important decision yo... | What is the biggest most important decision yo... |
| **5648** | 5648 | Where should I start learning C? | Where should I start learning C ? |
| **401** | 401 | How do I motivate myself to wake up early? | How do I motivate myself to wake up early ? |
| **1471** | 1471 | What is the best age to tell your kid that he'... | What is the best age to tell your kid that he'... |
| **1205** | 1205 | What are the reasons why wars happen? | What are the reasons why wars happen ? |

## POS-Tagging

Now the text is clean of special characters and unnecessary punctuations. We move to the next phase and obtain the **pos tags** for every sentence

In [51]:
```
def get_pos_tags(text):
    return pos_tag(text.split())
```

In [52]:
```
# Sample test
get_pos_tags(clean_text("What does Balaji Vishwanathan think about Modi's new cu
rrency Idea?"))
```

Out[52]:
```
[('What', 'WP'),
 ('does', 'VBZ'),
 ('Balaji', 'NNP'),
 ('Vishwanathan', 'NNP'),
 ('think', 'VBP'),
 ('about', 'IN'),
 ("Modi's", 'NNP'),
 ('new', 'JJ'),
 ('currency', 'NN'),
 ('Idea', 'NNP'),
 ('?', '.')]
```

**Once we obtain the POS-tag for each word in the sentence, we will append that tag to the word**

Once the tags are obtained we will convert all the words to lowercase

In [53]:
```
def append_pos_tags(text):
    pos_tagged_list = get_pos_tags(text)
    tag_map = {}
    tag_map['N'] = wn.NOUN
    tag_map['J'] = wn.ADJ
    tag_map['V'] = wn.VERB
    tag_map['R'] = wn.ADV

    lemma_text = ' '.join([lemmatizer.lemmatize(word.lower(), tag_map.get(tag
[0], wn.NOUN)) for word, tag in pos_tagged_list])
    return lemma_text.strip()
```

In [54]:
```
# Sample test
append_pos_tags(clean_text("What does Balaji Vishwanathan think about Modi's new
currency Idea?"))
```

Out[54]:
```
"what do balaji vishwanathan think about modi's new currency idea ?"
```

In [55]:
```
# Apply this transformation on all the records in data
df['tagged_sentence'] = df['clean_sentence'].apply(append_pos_tags)
df.sample(5)
```

Out[55]:

| | sentenceId | sentence | clean_sentence | tagged_sentence |
|---|---|---|---|---|
| **1974** | 1974 | When will I know I found the one? | When will I know I found the one ? | when will i know i find the one ? |
| **7176** | 7176 | Why were the polls so inaccurate in the 2016 e... | Why were the polls so inaccurate in the 2016 e... | why be the poll so inaccurate in the 2016 elec... |
| **6505** | 6505 | What common mistake do people make when choosi... | What common mistake do people make when choosi... | what common mistake do people make when choose... |
| **3105** | 3105 | What is the best earphones available under Rs.... | What is the best earphones available under Rs ... | what be the best earphone available under r . ... |
| **6937** | 6937 | How much does it cost (yearly) to do a MS in C... | How much does it cost yearly to do a MS in Can... | how much do it cost yearly to do a m in canada... |

## Data transformation

Convert the text into Tf-Idf vectors

In [70]:
```
vectorizer = TfidfVectorizer(max_df=0.95, ngram_range=[1,1])
```

In [71]:
```
tfidf_mat = vectorizer.fit_transform(df['tagged_sentence'])
tfidf_mat.shape
```

Out[71]: (9534, 2990)

In [58]:
```
# Checking if we have any rows with all values as zeros
# This situation might occur if any sentence only has infrequent words which we have dropped during vectorization
np.where(~tfidf_mat.todense().any(axis=1))[0]
```

Out[58]: array([], dtype=int64)

In [59]:
```
with open("vectorizer.pkl", "wb+") as fp:
    pickle.dump(file=fp, obj=vectorizer)
```

**We know that the Tf-Idf representation is sparse, let us use some dimensionality reduction technique to reduce the dimensions and obtain a dense representation of data**

## Dimensionality Reduction

- Use PCA to reduce the dimensions

In [74]:
```
pca = PCA(n_components=0.95)
pca_vectors = pca.fit_transform(tfidf_mat.todense())
```

In [75]:
```
pca_vectors.shape
```

Out[75]: (9534, 1256)

**Due to memory constraints Fixing the number of components to 250. Since the Frobenius norm is high we can increase the components count**

In [102]:
```
with open("pca_vectors.pkl", "wb+") as fp:
    pickle.dump(file=fp, obj=pca_vectors)
```

In [103]:
```
with open("pca_vectors.pkl", "rb") as fp:
    pca_vectors = pickle.load(file=fp)
```

## Similarity Calculation

- I'll be using cosine similarity as my metric

In [61]:
```
def cosine_similarity(A, B):
    return np.dot(A, B) / (np.linalg.norm(A)*np.linalg.norm(B))
```

In [119]:
```
# This line of code calculates the similarity between each records and every ot
her record.
# But for each input sentence we require only top 3. So no point in wasting mem
ory for the rest


# similarity_matrix = cdist(pca_matrix, pca_matrix, cosine_similarity)
```

**I'll be implementing a Heap based solution where in for each sentence we maintain a heap to keep track of top-3 similar sentences while iterating over the whole dataset**

In [62]:
```
def add_to_results(results, index, inner_index, score, TOP_K):
    heap = results[index]
    # Pushing the similarity score and inner_index index of the sentence
    heapq.heappush(heap, (score, inner_index))
    # Since we only need to store K=3 similar sentences, pop other sentences wit
h lesser scores
    if len(heap) > TOP_K:
        _ = heapq.heappop(heap)
    results[index] = heap
    return results

def get_top_k_similar_sentences(matrix, func = cosine_similarity, TOP_K = 3):
    results = [[] for _ in range(matrix.shape[0])]
    # Iterating over all sentences
    for index in tqdm(range(matrix.shape[0])):
        # Each Input sentece in PCA based representation
        vector = matrix[index]
        # Initializing empty list to keep track of similar sentences
        heap = []
        # Iterating over rest of the sentences
        for inner_index in range(index, matrix.shape[0]):
            # checking if outer loop index and inner loop index are not equal
            if index != inner_index:
                # calculate cosine similarity between two vectors
                score = func(vector, matrix[inner_index])
                # Adding similarity scores to the results
                results = add_to_results(results, index, inner_index, score, TOP
_K)
                # Since cosine similarity score is symmetric
                results = add_to_results(results, inner_index, index, score, TOP
_K)
        # Sorting the array in descending order before storing into results
        results[index] = sorted(results[index], key=lambda x : x[0], reverse=Tru
e)
    return results
```

In [79]:
```
TOP_K = 3
results = get_top_k_similar_sentences(matrix=pca_vectors,
                                      func=cosine_similarity,
                                      TOP_K=TOP_K)
```

```
100%|██████████████████████████████████████████████████████████████████████
███| 9534/9534 [5:38:57<00:00,  2.13s/it]
```

In [ ]:
```

```

In [122]:
```
with open("results.pkl", "wb+") as fp:
    pickle.dump(file=fp, obj=results)
```

In [63]:
```
with open("results.pkl", "rb") as fp:
    results = pickle.load(file=fp)
```

## Evaluating Results

In [148]:
```
def print_sample_results(df, results, index):
    print("Input: {}) {}".format(index, df.loc[index, "sentence"]))
    print("\nSimilar Sentences: ")
    for i, tup in enumerate(results[index]):
        if tup[0] > 0.5:
            print("{}) {} {:.2f}".format(tup[1], df.loc[tup[1], "sentence"], tu
p[0]))
```

In [135]:
```
for _ in range(10):
    index = np.random.randint(9533)
    print_sample_results(df, results, index)
    print("\n====================================\n")
```

```
Input: 8631) Is downloading from torrent is still illegal in India?

Similar Sentences:
7568) Is downloading from torrent is still illegal in India? 1.00
3974) Is downloading from torrent illegal in India? 0.91
5363) Is downloading from torrent illegal in India? 0.91

====================================

Input: 1460) Which is the coldest country?

Similar Sentences:
1797) Which is the coldest country? 1.00
2008) Which is the coldest country? 1.00
3716) Which is the coldest country? 1.00

====================================

Input: 3982) Can any state secede from United States?

Similar Sentences:
3143) Can any state secede from United States? 1.00
1096) Can any state secede from United States? 1.00
1839) Can any state secede from United States? 1.00

====================================

Input: 9009) What is your favourite poem and why?

Similar Sentences:
5939) What is your favourite poem and why? 1.00
7848) Which is your favourite poem and why? 0.93
8292) Which is your favourite poem and why? 0.93

====================================

Input: 7644) Which laptop or notebook can I buy under 20k?

Similar Sentences:
5028) Which laptop or notebook can I buy under 20k? 1.00
6712) Which laptop or notebook can I buy under 20k? 1.00
9067) Which laptop or notebook can I buy under 20k? 1.00

====================================

Input: 1666) What are the best hotels in Rajasthan?

Similar Sentences:
1596) What are the best hotels in Rajasthan? 1.00
6363) What is the best hotel in Rajasthan? 1.00
6833) What are the best hotels in Rajasthan? 1.00

====================================

Input: 2579) Has there ever been a conflict of orbital paths between two satel
lites that were going to crash into each other?

Similar Sentences:
5704) Why don't satellites crash into each other? 0.63
7346) Why don't satellites crash into each other? 0.63
8544) Why don't satellites crash into each other? 0.63

====================================

Input: 3160) What is the First Amendment's purpose?

Similar Sentences:
8067) What is the First Amendment's purpose? 1.00
```

```
3300) What is the First Amendment? 0.79
1661) What is the First Amendment? 0.79

====================================

Input: 3944) Why do we always see the same side of the Moon from Earth?

Similar Sentences:
1959) Why do we always see the same side of the Moon from Earth? 1.00
1448) Why do we always see the same side of the Moon from Earth? 1.00
4095) Why do we always see the same side of the Moon from Earth? 1.00

====================================

Input: 3666) When I get paid by Paytm by someone how do I transfer it to my ba
nk account?

Similar Sentences:
918) How do I transfer Paytm balance to bank account? 0.62
1144) How do I transfer Paytm balance to bank account? 0.62
8433) How do I transfer Paytm balance to bank account? 0.62

====================================
```

## Appending results to dataframe in required format

In [137]:
```
similar_indices = [0 for _ in range(len(results))]
similar_sentence_scores = [0 for _ in range(len(results))]
for key, row_results in enumerate(results):
    similar_indices[key] = ', '.join([str(row_result[1]) for row_result in row_
results if row_result[0] >= 0.5])
    similar_sentence_scores[key] = ', '.join(["{:.2f}".format(row_result[0]) fo
r row_result in row_results if row_result[0] >= 0.5])
```

In [138]:
```
df['Similar Sentences'] = similar_indices
df['Similarity Score'] = similar_sentence_scores
```

In [139]:
```
df['has_similar'] = df['Similarity Score'] != ''
```

## Writing results back to Disk

In [140]:
```
writer = pd.ExcelWriter(os.path.join(DATA_DIR, LEXICAL_OUTPUT_FILE))
df.loc[df['has_similar'], ['sentenceId', 'sentence', 'Similar Sentences', 'Simi
larity Score']].to_excel(writer, sheet_name="Sentence Similarities", index=Fals
e)
df.loc[~df['has_similar'], ['sentenceId', 'sentence']].to_excel(writer, sheet_n
ame="No Similar Sentence", index=False)
writer.save()
writer.close()
```

In [ ]:

In [ ]:

In [ ]: